# 基于混沌系统的图像可逆信息隐藏算法

## 1 算法描述

### 1.1 基本算法描述

为了叙述上的方便，我们将以 512×512 像素大小的灰度图像Lena.bmp 为例，将基于直方图的图像可逆信息隐藏算法描述如下：

（1）分析原始图像直方图。读入原始图像，并绘制该图像的直方图，如图 1 所示。在该直方图内最大值点对应的像素值为 156，说明在原始图像内该像素值的像素点个数最多；其左端值为 0~28和其右端值为245~255的像素值均为0，说明在原始图像中不存在这些像素值的像素点。



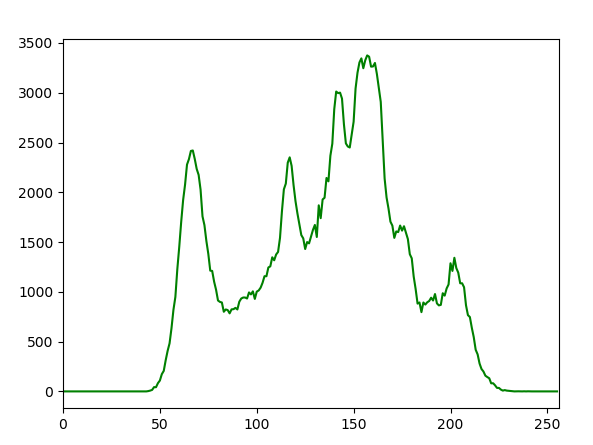


图 1 原始图像及灰度直方图

(2)直方图处理。对图像进行扫描，如果扫描到的像素点其值为[29，155]，则将其值减去 1。扫描完成后，在原图内不再存在像素值为 155 的像素点。

(3)嵌入信息。再次对图像进行扫描，如果扫描到的像素点其值为 156，则可以在该点隐藏 1 位信息。隐藏规则为，如果待嵌入点为 0，则该点保持不变；如果待嵌入点为 1，则将该点值减 1。至此，信息隐藏完成。

(4)提取信息。对隐藏了信息的图像进行顺序扫描，如果扫描到的像素点值为 156，则提取一个信息位 0，如果扫描到的像素点值为 155，则提取一个信息位 1，扫描过程完成即完成隐藏信息的提取。

(5)恢复原始图像。再次扫描图像，将在图像内扫描到的像素点值在[28 155]之间的点加 1，即完成图像的复原。

该算法利用图像中最大像素值点来实现信息嵌入，因此，信息嵌入最大量即为最大像素值像素点的个数，从图像直方图可以看出，该点在图像中的数量有限，因此可以嵌入信息的数据量并不大。在嵌入时，该算法只是将原始信息隐藏到对应的峰值点，嵌入完成后会在靠近该峰值点附近形成波谷，通过统计分析很容易实现隐藏信息的提取，因此，算法的安全性不高。改进算法提出利用峰值点两侧的连续多个零值点实现大容量的信息嵌入策略，同时采用混沌系统对秘密信息进行加密从而保证隐藏信息的安全性。

### 1.2 基于混沌系统的算法

通过对大量图像的直方图分析得知，大多数图像的直方图如图1所示，直方图内左右两侧端点均存在连续多个值为 0 的点。我们通过在峰值点和这些点之间建立对应的关系，实现将经过混沌加密后的秘密信息嵌入到原始图像内。

以512×512 像素大小的灰度图像 Lena.bmp 为载体图像，将基于混沌系统的图像可逆信息隐藏算法描述如下：

#### 1.2.1 秘密信息的处理

在信息嵌入时，先采用混沌系统对秘密信息进行混沌置乱，在提取过程中，采用与嵌入时相同的混沌初始值对提取出的秘密信息进行逆置乱。

被广为应用的混沌是Logistic混沌系统，它的改进形式为:

该系统对初始值极为敏感，初始值稍微不同，就会出现完全不同的结果。改进算法提出的可逆信息隐藏算法采用该混沌系统对秘密信息进行加密、解密。

用混沌系统产生一个长度为n(此处n为需要置乱的待隐藏信息的数列长度)的数列 CI，将该数列进行排序，从而得到一个升序数列 CIS。通过计算数列 CIS 中的每个数据在数列CI 中的下标，得到一个数列 I。应用公式为:

式中：函数FI的功能为找出参数在数列CI中的下标。

嵌入隐藏信息时，先将待隐藏信息数列 W 按照数列 I 进行置乱排序得到数列 WO，应用公式为

然后，将 WO 嵌入到载体图像中。

提取隐藏信息时，在含隐藏信息的图像中提取隐藏信息数列 WO \* ，然后将 WO \* 进行逆置乱，应用的公式为

此时，得到正常排序的隐藏信息。

应用上述混沌处理过程保证了嵌入过程的安全性和可靠性。

#### 1.2.2 嵌入隐藏信息

(1)读取载体图像，绘制出其直方图，找出该直方图的峰值点Max及其左侧的连续零值点Li ,i∈[0,Max]，右侧连续零值点R i ,i∈(Max,255)。

(2)为了提高信息的隐藏容量和嵌入信息后图像的透明性，针对峰值点，在直方图左右两侧选择连续零值点进行匹配嵌入，每个峰值点能嵌入的隐藏信息位数En、所需左右两侧连续零值点个数总和 Sn、所需左侧连续零值点个数 Ln、所需右侧连续零值点个数 Rn之间的关系如表 1 所示。

表 1 嵌入信息位数及零值点关系

|  |  |
| --- | --- |
| 最大像素点嵌入的隐藏信息位数En | 连续零值点个数 Ln 及 Rn |
| 1 | Ln+Rn=1, |
| 2 | Ln+Rn=3, |
| 3 | Ln+Rn=7, |
| 4 | Ln+Rn=15, |
| 5 | Ln+Rn=31, |

(3) 隐藏信息时，顺序扫描图像，如果遇到峰值点 Max，则从经过混沌置乱后的信息序列中选取En个比特位，如果选取的 En 个比特位均为 0，则像素值保持不变；否则，将进行像素值加减完成信息嵌入。

按照上述步骤顺序扫描图像内所有像素点，即可完成嵌入。在实际操作中，可以先选取一个峰值点对应 4 位嵌入信息完成嵌入，嵌入完成后根据所得到的 PSNR 值和实际需要的嵌入容量对嵌入位数进行进一步的修正。

#### 1.2.3 提取隐藏信息及恢复原始图像

从改进算法的嵌入算法可以看出，信息提取过程和恢复原始图像过程即为信息嵌入过程的逆过程。从嵌入过程可以看出，该算法可以实现嵌入信息的完全提取和原始图像的无损恢复。

在信息提取及原始图像的恢复过程中需要注意以下问题：

(1) 在信息提取过程中，应根据嵌入时实际选择的连续零值点个数 Sn、左侧连续零值点个数 Ln、右侧连续零值点个数Rn进行信息提取，采用与嵌入过程不匹配的值将无法提取正确信息。

(2) 在完成信息提取后，提取出的隐藏信息为经过混沌置乱后的信息，需要对信息进行逆置乱，根据 1.2.1 小节中介绍的混沌系统及与加密时所对应的混沌初始值，实现隐藏信息的逆置乱提取出原始隐藏信息。

## 2 实验测试及分析

我们应用一些灰度图像为例进行仿真实验，仿真结果显示海禁算法能够在嵌入大量信息的同时保持较高的峰值信噪比，我们选取一些典型的512×512 像素大小的灰度图像进行分析。

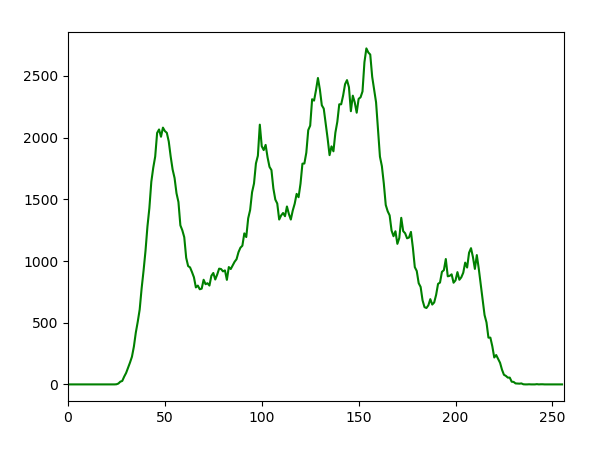
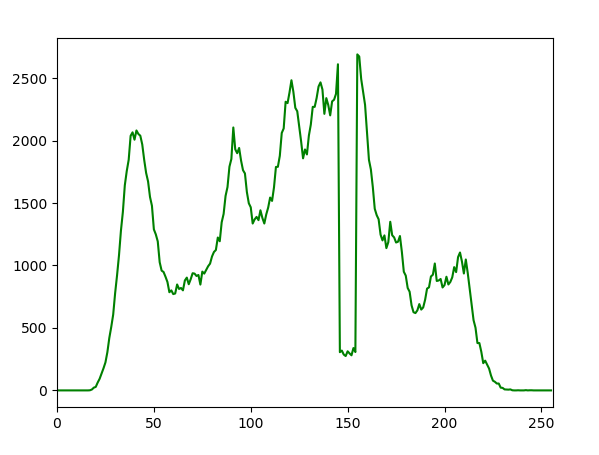
### 2.1 容量及透明性分析

可逆信息隐藏算法的两个重要指标是透明性及嵌入容量。透明性要求信息隐藏算法在完成隐藏信息嵌入后的图像与原始图像计算具有较高的峰值信噪比，即嵌入隐藏信息前后图像不发生明显的失真。嵌入容量用来衡量一个算法能够在图像载体内嵌入信息量的大小。通常情况下，如果要保持较高的峰值信噪比则要求嵌入的信息量越少越好；另一方面，如果需要保持比较高的嵌入容量，则会对图像修改较多，造成一定的失真，即峰值信噪比会降低。因此，峰值信噪比和嵌入容量是一对矛盾。

改进算法提出在直方图两侧分别修改零值点以完成隐藏信息的嵌入，保证了算法既能够嵌入大容量的信息，又能够保持较高的峰值信噪比。应用改进算法，在峰值点左右两侧选取 15个像素点参与信息嵌入，一些典型图像嵌入前后的图像及灰度直方图如图 2 所示。从嵌入前后的图像可以看出，图像在嵌入完成后并未发生明显失真。

(a)原始图像 (b)含隐藏信息

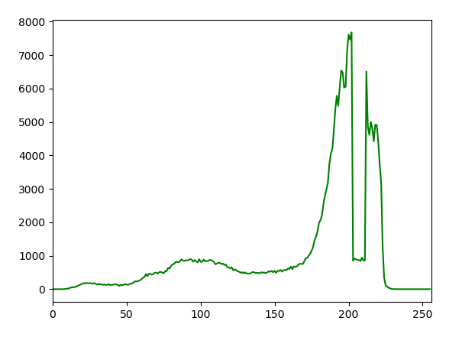
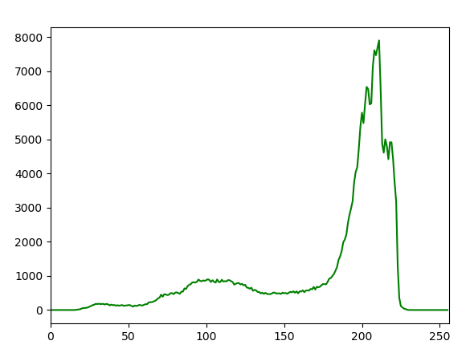
 

(c)原始直方图 (d)含隐藏信息

图 2 Lena 图像

(a)原始图像 (b)含隐藏信息

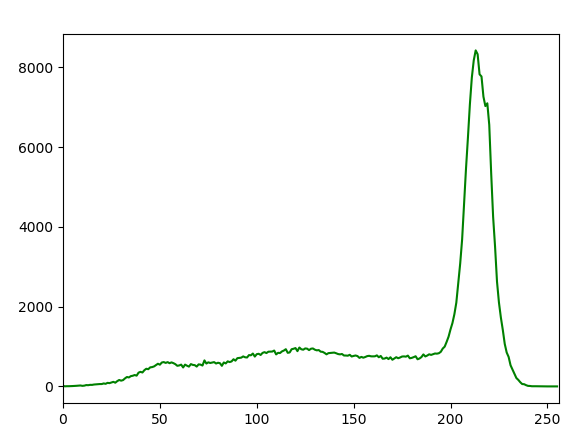
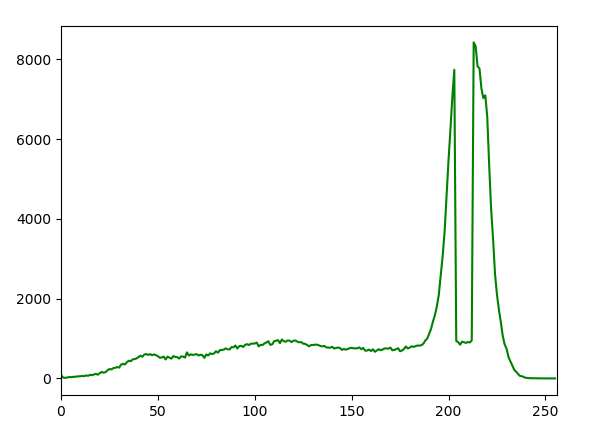


(c)原始直方图 (d)含隐藏信息

图3 Plane 图像

(a)原始图像 (b)含隐藏信息

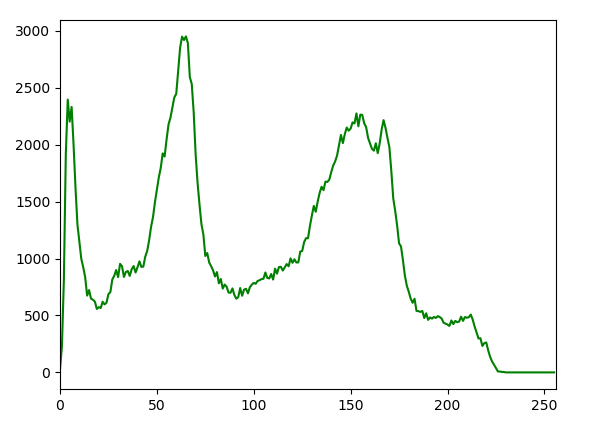
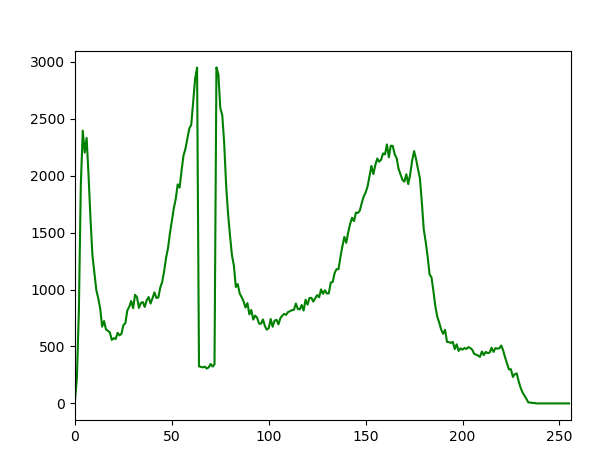
 

(c)原始直方图 (d)含隐藏信息

图4 Dollar图像

(a)原始图像 (b)含隐藏信息

(c)原始直方图 (d)含隐藏信息

图5 Peppers图像

对上述图像在峰值点嵌入了3个比特位，采用改进算法实现所得图像的对应PSNR值与采用原始算法所得图像的 PSNR 值进行比较，结果如图6所示。从图中可看出，在嵌入等量信息的情况下，采用改进算法得到的 PSNR 值均要好于原始算法所得到的 PSNR 值。仿真结果证明，经过改进算法嵌入隐藏信息的图像透明性更好。

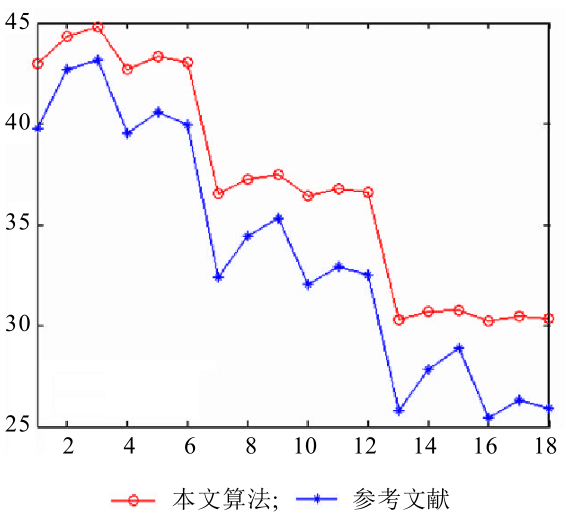


图6 PSNR 值比较

信息的嵌入容量指在载体图像内能够嵌入的秘密信息位数。在本算法中，采用峰值点及其两侧连续零值点之间的关系完成信息的嵌入，因此可以嵌入的信息量 S 为

式中：NMax——峰值点像素个数，En——每个峰值像素点对应的嵌入位个数。

大量仿真实验表明，当 En 值为 4 时，本算法仍可以取得比较好的峰值信噪比，部分图像当En取值为5 时，仍旧具有较高的峰值信噪比。因此，本算法的嵌入容量为基本算法的 4 至 5 倍左右。

### 2.2 安全性分析

我们采用了混沌系统对要隐藏的信息进行加密。混沌系统具有对初始值敏感的特性，初始值的微小改变都会使系统偏离原来的解，同时系统不具有长期的可预测性。如图 7 所示，实线的初始值为 0.8，虚线的初始值为 0.80000001，虽然初始值只差了 0.000 000 01，但经过大约 30 次的迭代后，两条曲线开始分散，以至完全独立，失去相关性。由此可见 Logistic混沌映射对初始值极为敏感，初始值稍微不同，就会迅速变成完全不同的状态，同时也清楚地表明了混沌映射的非稳态性，迭代次数足够大时，混沌序列将表现出非周期性。因此，如果试图在不知道混沌初始值的情况下解密加密信息，则此时的密钥空间为(-1,1)内的所有有理数，密钥空间为无限大。这些特点使得混沌系统的加密具有较高的安全性。

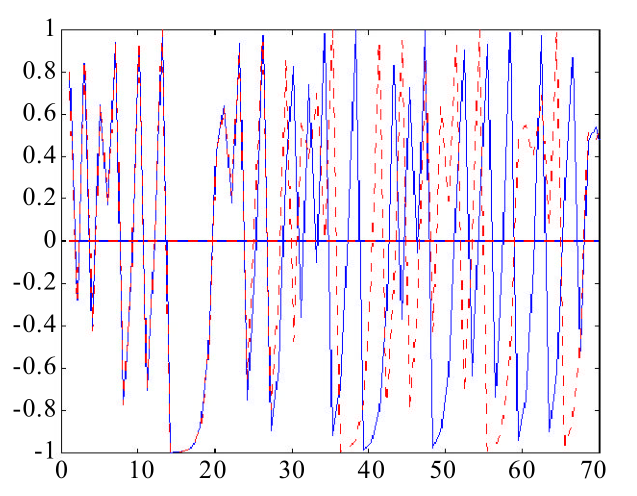


图7 混沌序列

实验仿真结果显示，应用混沌序列对图像进行加密，具有较高的安全性。即使攻击密钥与真实密钥具有微小差异，所解密图像也会与明文图像完全不同。

## 3 实验代码

### 3.1 图片分析代码：

import cv2 as cv

import matplotlib.pyplot as plt

import numpy as np

from pylab import \*

src = cv.imread("lena512.bmp")

#cv.imshow("q",src)

h,w,ch = np.shape(src)

gray = cv.cvtColor(src,cv.COLOR\_BGR2GRAY)

#cv.imshow("gray",gray)

pre=0

ap=[]

hest = np.zeros([256],dtype = np.int32)

for row in range(h):

for col in range(w):

pv = gray[row,col]

hest[pv] +=1

if col!=0:

ap.append(int(pre)-int(pv))

pre=pv

print(max(hest))

ap=np.array(ap)

print(np.sum(ap== -1))

akb=[]

ske=[]

for i in range(-255,255):

akb.append(i)

ske.append(np.sum(ap==i))

fig=plt.figure()

ax=fig.gca()

xlim(-20,+20)

plt.plot(akb,ske,color='green')

plt.scatter(akb,ske,marker='\*')

src = cv.imread("aaa.bmp")

#cv.imshow("q",src)

h,w,ch = np.shape(src)

gray = cv.cvtColor(src,cv.COLOR\_BGR2GRAY)

#cv.imshow("gray",gray)

pre=0

ap=[]

hest = np.zeros([256],dtype = np.int32)

for row in range(h):

for col in range(w):

pv = gray[row,col]

hest[pv] +=1

if col!=0:

ap.append(int(pre)-int(pv))

pre=pv

print(max(hest))

ap=np.array(ap)

print(np.sum(ap== -1))

akb=[]

ske=[]

for i in range(-255,255):

akb.append(i)

ske.append(np.sum(ap==i))

xlim(-20,+20)

plt.plot(akb,ske,color='red')

plt.scatter(akb,ske)

plt.show()

### 3.2 信息隐藏代码

import cv2 as cv

import matplotlib.pyplot as plt

import numpy as np

from pylab import \*

import random

import scipy.misc

src = cv.imread("dollar.bmp")

#cv.imshow("q",src)

h,w,ch = np.shape(src)

gray = cv.cvtColor(src,cv.COLOR\_BGR2GRAY)

#cv.imshow("gray",gray)

'''

list\_W:list\_init每个元素在list\_pai的索引位置

list\_pai:排序的混沌序列

list\_init:初始混沌序列

'''

init=0.633

list\_hun=[]

len\_cipher=10

for i in range(len\_cipher):

init=round(1-init\*init\*2,4)

list\_hun.append(init)

list\_pai=sorted(list\_hun)

list\_W=[ list\_hun.index(i)for i in list\_pai]

ap=[]

hest = np.zeros([256],dtype = np.int32)

for row in range(h):

for col in range(w):

pv = gray[row,col]

hest[pv] +=1

if int(pv)<212:

ap.append(pv-8)

elif int(pv)==212:

ap.append(212-random.randint(0,8))

else:

ap.append(pv)

ap=np.array(ap)

ap=np.reshape(ap,(512,512))

cv.imwrite("dollaraaa.bmp",ap)

plt.plot(hest,color = "r")

plt.xlim([0,256])

plt.show()

cv.waitKey(0)

cv.destroyAllWindows()

#statistics()

'''

进行直方图分析

'''

src = cv.imread("peppers.bmp")

#cv.imshow("q",src)

h,w,ch = np.shape(src)

gray = cv.cvtColor(src,cv.COLOR\_BGR2GRAY)

#cv.imshow("gray",gray)

pre=0

ap=[]

hest = np.zeros([256],dtype = np.int32)

for row in range(h):

for col in range(w):

pv = gray[row,col]

hest[pv] +=1

plt.plot(hest,color = "g")

plt.xlim([0,256])

plt.show()